

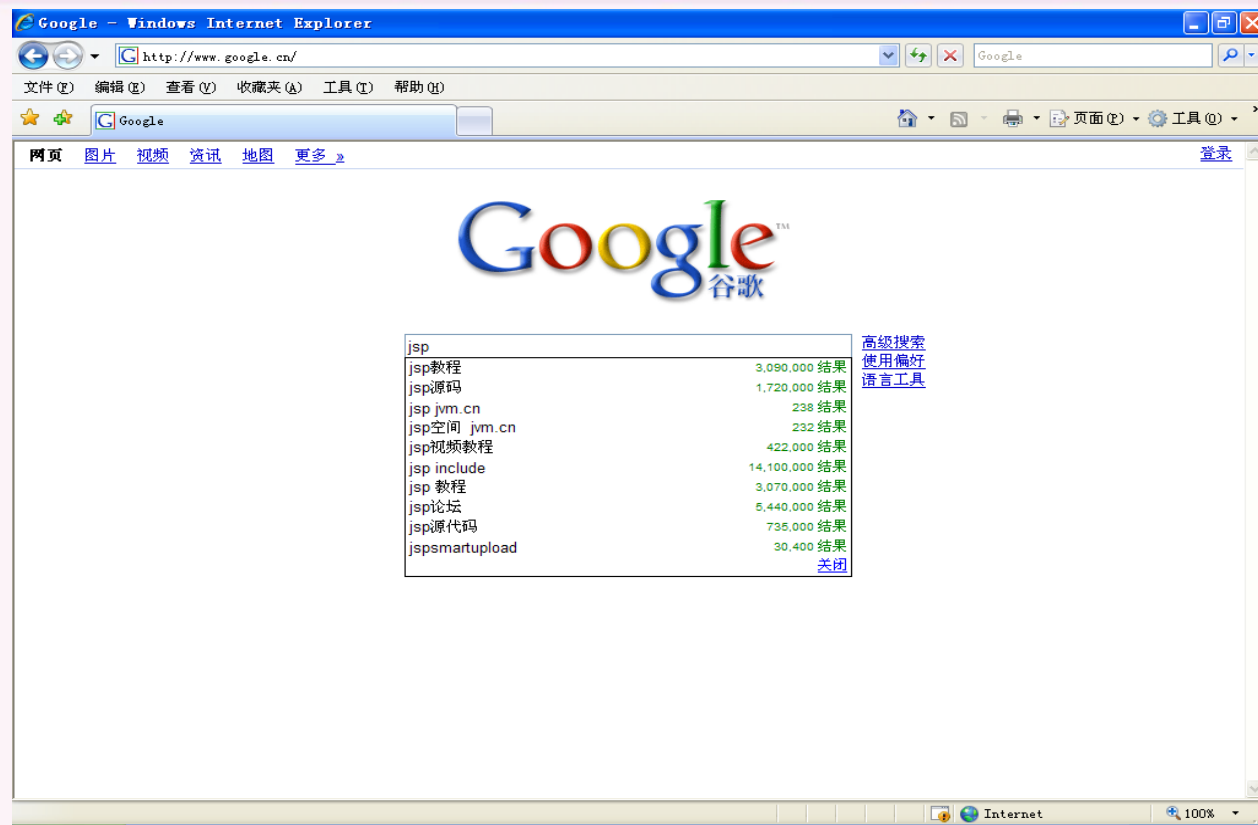
WEB

第 10 章 Web2.0 开发——Ajax

本章目录

- 10.1 Ajax 的定义及特点
- 10.2 Ajax 的工作原理
- 10.3 Ajax 开发调试技巧
- 10.4 Ajax 开发实例
- 10.5 第三方 Ajax 高级框架
- 10.6 本章小节

引例



Ajax 在 Google 中的运用

10.1 Ajax 的定义及特点

- 全称 **Asynchronous JavaScript and XML**
- 中文名为异步 JavaScript 和 XML
- Ajax 并不是一门新的语言，它是通过巧妙组合 HTML、DHTML、JavaScript 以及 DOM 等技术形成的一种新的 Web 应用。

Ajax 技术的特点

1. 操作上的友好性

用户可以在不改变当前网页内容的情况下，得到用户所需要的内容。

2. 减轻服务器的负担

Ajax 的运行方式则按照用户的需要在当前页面显示数据，这样就大大地减轻了服务器处理动态网页的负担，同时也大大增加了用户取得数据的速度。

3. 标准化和广泛支持

Ajax 的支撑技术均是国际上的一些标准技术，比如 JavaScript、XML 等，因此使程序员能够非常方便地开发出基于 Ajax 的 Web 应用。

10.2 Ajax 的工作原理

Ajax 的工作原理是通过在浏览器和服务器之间增加一个中间层，进而实现用户请求与服务器响应的异步通信。

Ajax 的核心是 JavaScript 对象 XMLHttpRequest
Ajax 引擎就是借助于 JavaScript 程序，通过调用 XMLHttpRequest 对象的属性和方法来与服务器端进行交互。

同步通信示意图

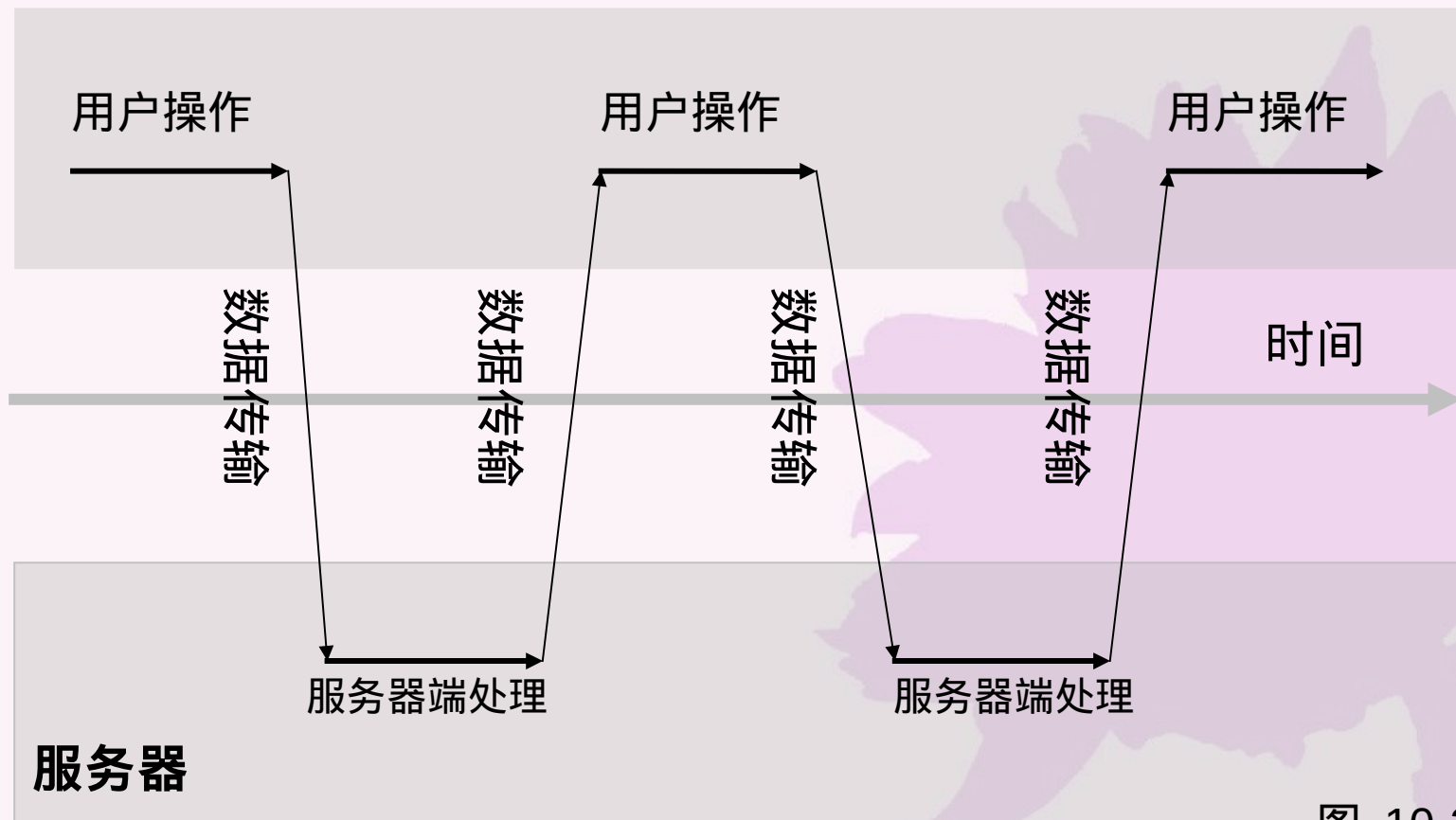


图 10-2

异步通信示意图

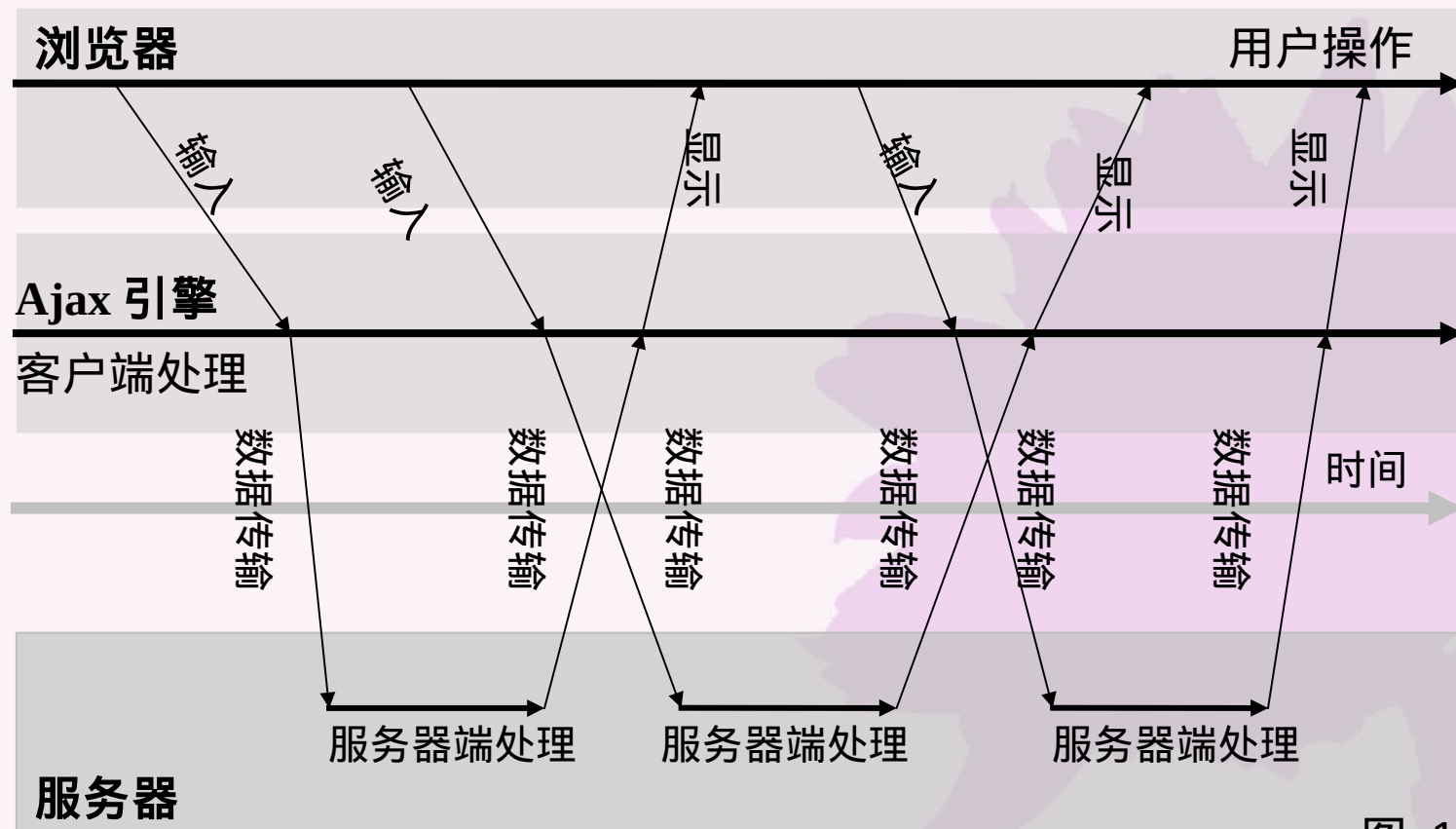
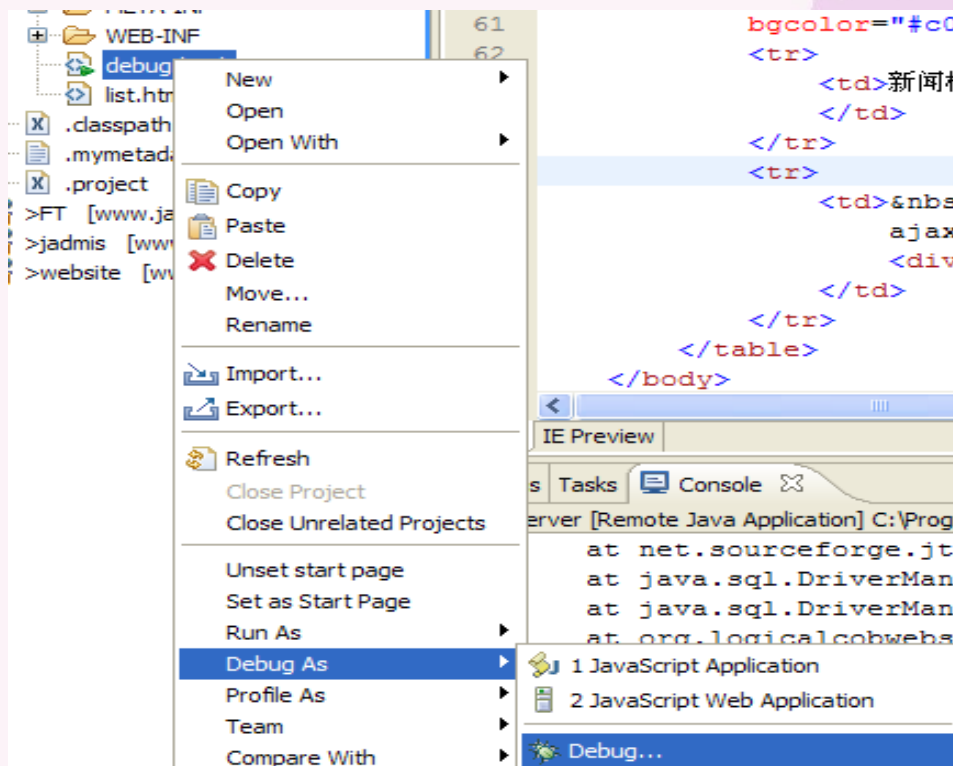


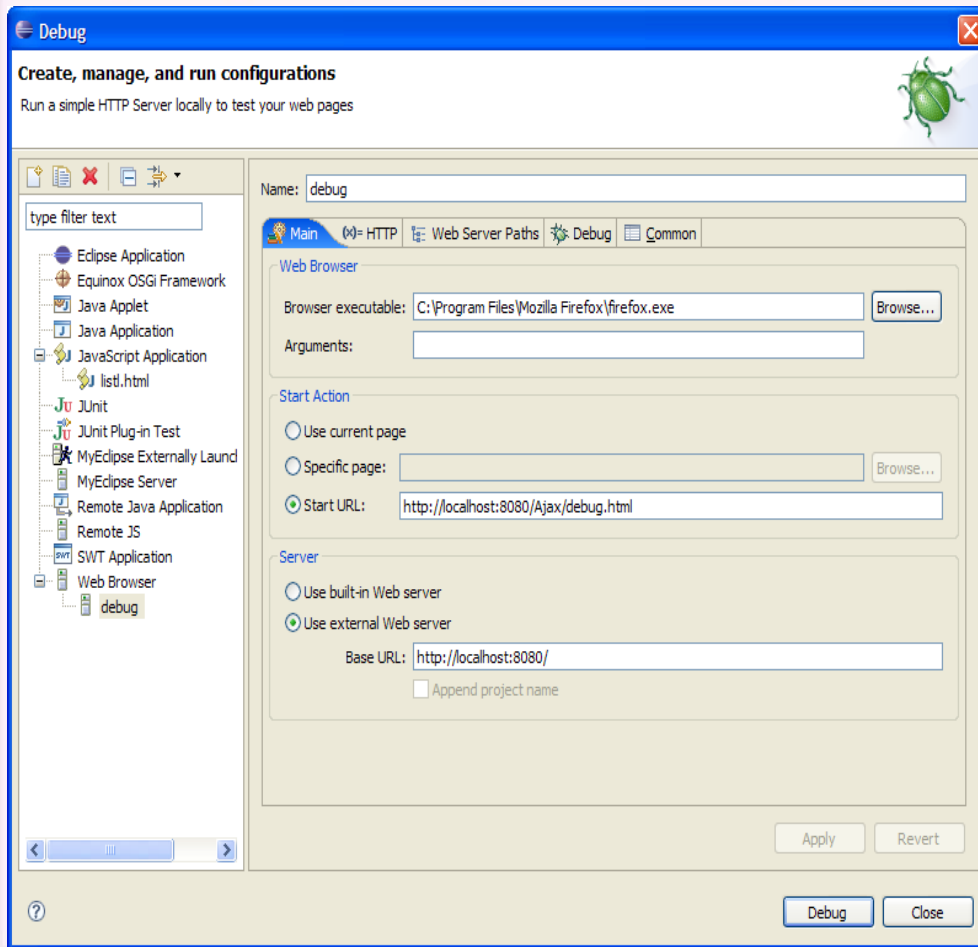
图 10-3

10.3 Ajax 开发调试技巧

- 打开项目，启动 Tomcat 服务器
- 右击要调试的文件【 Debug As 】→【 Debug... 】



在图中选择【 WebBrowser 】，然后单击左上角的小按钮，系统会弹出详细配置信息。



- “Name” 是指新建浏览器的名称。

- “Browser executable” 输入框需填入 Firefox 的安装路径

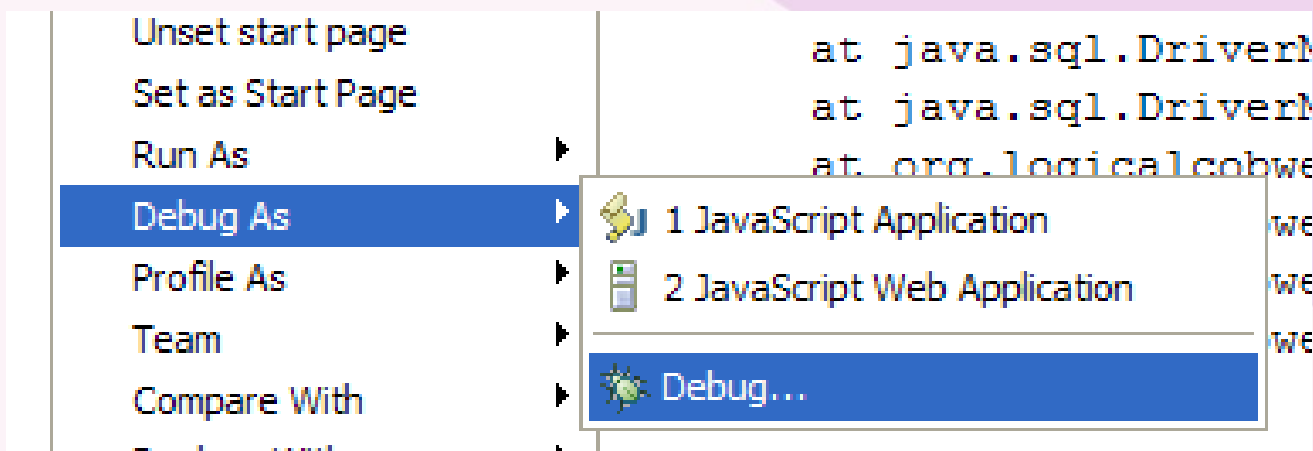
- Start Action 选项应选择“ Start URL”，指待调试文件的 URL 路径。

- Server 选项选择“ Use external Web server”，并在“ Base URL” 中填入

http://localhost:8080

断点调试

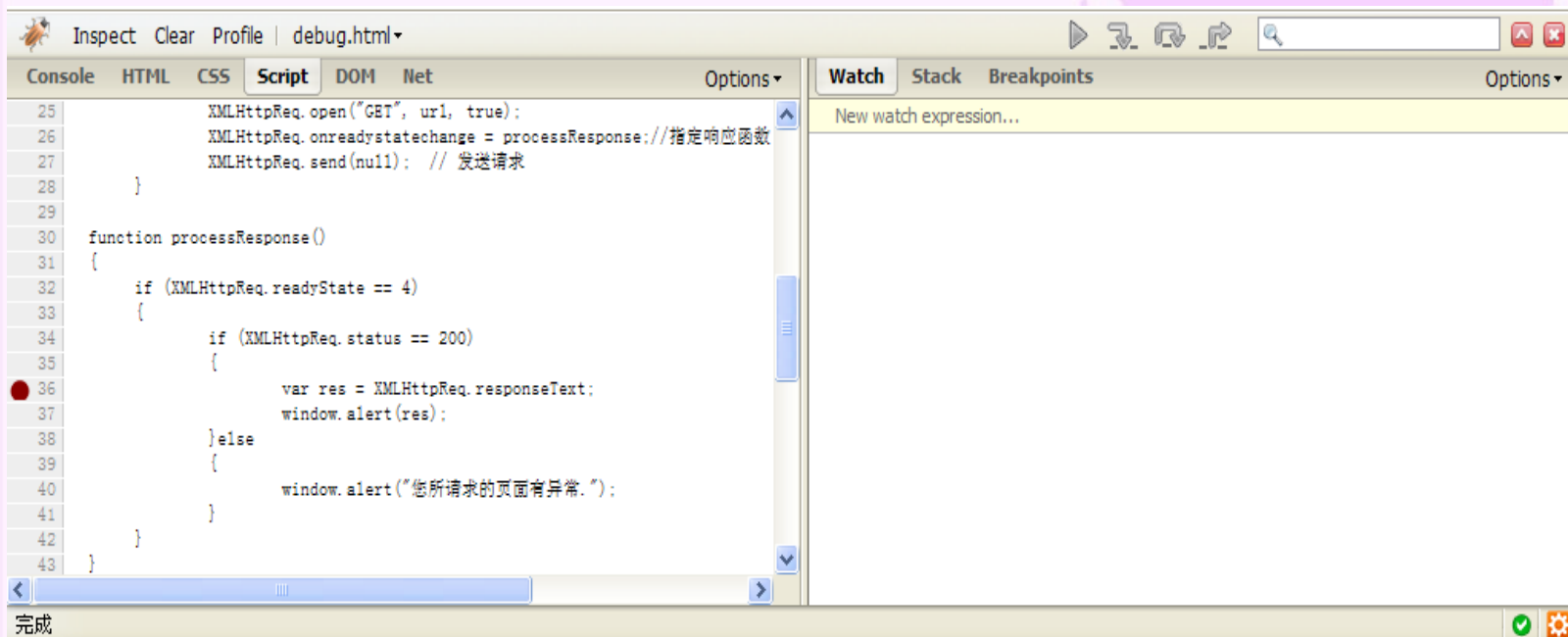
需要说明的是，右击 debug.html 后，如图 10-6 所示，在弹出的菜单里不能选择【Javascript Web Application】和【JavaScript Application】，这两个选项分别适用于单个页面和 JS 文件。所以，我们还是要选择【Debug】选项，然后按照上面介绍的方法启动调试。



启动 Firefox 后，注意 Firefox 的右下角多了两个小图标，如图 10-7 所示：

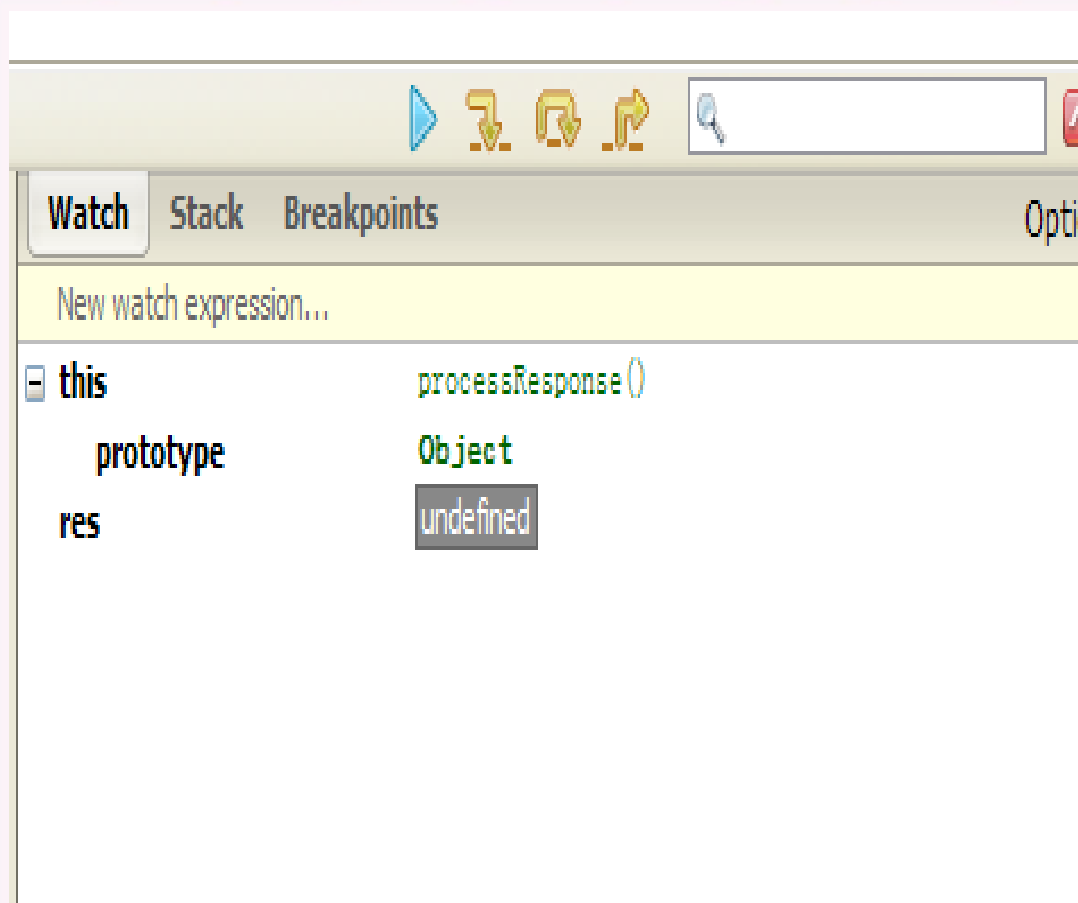


点击该小图标，显示打开调试控制台，如图 10-8 所示：



单击【Script】选项卡，在需要设置断点的代码行的标记栏上双击即产生一个断点，当程序运行到这行代码时，线程被挂起，如图 10-8 所示。

利用单步操作按钮，我们可以对程序进行单步执行并查看相关变量或对象的内容和状态。



1. “继续”按钮
2. “单步跳入”按钮
3. “单步跳过”按钮
4. “单步过滤”按钮

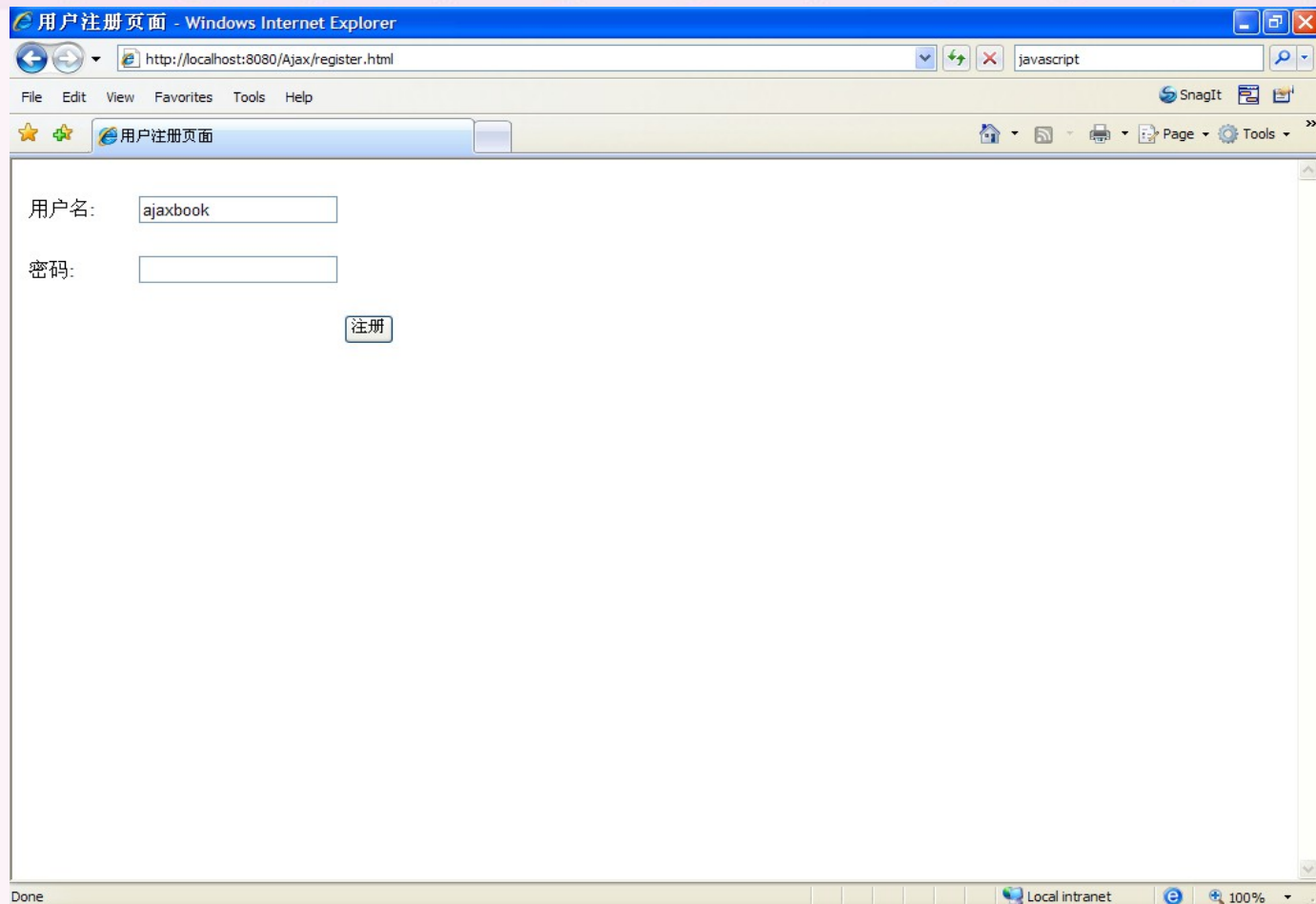
10.4 Ajax 开发实例

10.4.1 用户名唯一性检验

在 MySQL 数据库中新建一个数据库表，表名为 userinfo，为该表创建三个字段。userinfo 表建成后，向表中手动添加几条记录，如图 10-10 所示：

userid	username	userpwd
1	ajaxuser	ajaxuser
2	ajaxbook	ajaxbook

在 MyEclipse 中新建一个名为“ Ajax” 的 Web 项目，并在 Ajax 项目中新建一个注册页面 register.html，如图 10-11 所示：



在以上表单中的用户名输入框里需要预设一个 onblur 的事件处理，并设计一个单元格用于显示验证后的提示信息。

```
<tr>
<td width="82"> 用户名 :</td>
<td width="156"><input type="text" name="username"
id="username" onblur="sendRequest()"></td>
<td width="285" id="tip"></td>
</tr>
```

我们设计的思路是用户输入完用户名之后，鼠标一旦离开用户名输入框，输入框不再获得焦点，此时触发 blur 事件，因此我们可以在 onblur 事件处理中调用 sendRequest() 函数将用户名发送给服务器的程序进行验证并将返回的文本信息设置到“tip”单元格中。

下面在 <head></head> 标签中编写 Ajax 处理所涉及的 JavaScript 函数，首先是函数 createXMLHttpRequest()，

```
var XMLHttpReq;  
function createXMLHttpRequest() {  
    if(window.XMLHttpRequest) { //Mozilla 浏览器  
        XMLHttpReq = new XMLHttpRequest();  
    }  
    else if (window.ActiveXObject) { // IE 浏览器  
        try {  
            XMLHttpReq = new  
ActiveXObject("Msxml2.XMLHTTP");  
        } catch (e) {  
            try {  
                XMLHttpReq = new  
ActiveXObject("Microsoft.XMLHTTP");  
            } catch (e) {}  
        }  
    }  
}
```

XMLHttpRequest 的常用属性：

1.Onreadystatechange：指定XMLHttpRequest对象状态改变时的响应程序。

2.readyState：XMLHttpRequest对象的处理状态。

3.responseText：用于获取服务器的响应文本。

4.responseXML：用于获取服务器的响应XML。
。

5.status：服务器返回的状态值，表示服务器完成响应的状态。

接下来开发一个响应函数 processResponse()，代码如下：

```
function processResponse()
{
    if (XMLHttpRequest.readyState == 4)
    {
        if (XMLHttpRequest.status == 200)
        {
            var tip=XMLHttpRequest.responseText;

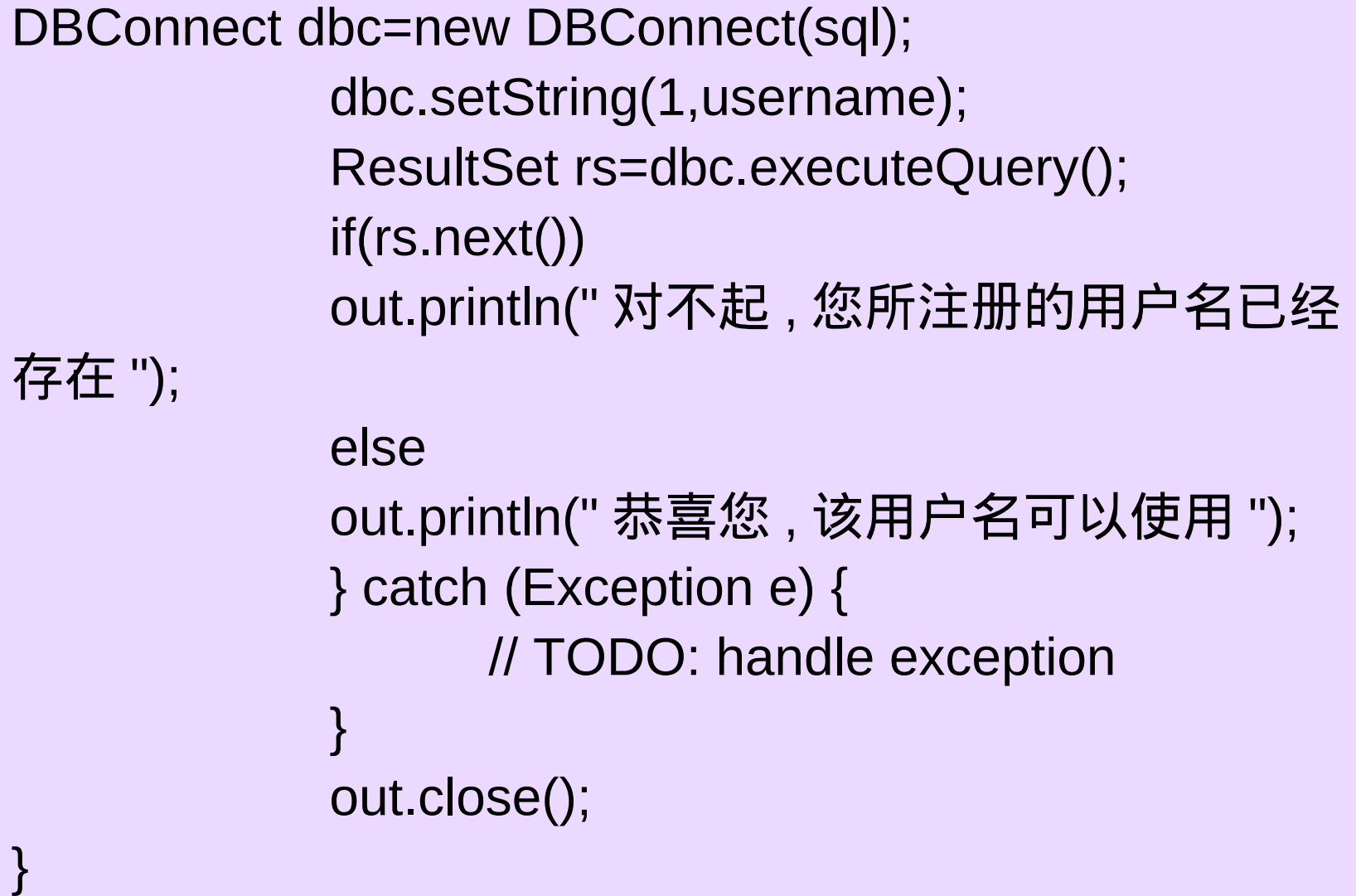
document.getElementById("tip").innerHTML="<font
color='blue'>" + tip + "</font>";
        }else
        {
            alert(" 您所请求的页面有异常 !");
        }
    }
}
```

有了 createXMLHttpRequest() 和 processResponse() 函数后，我们就可以完成 sendRequest() 函数了，代码如下：

```
function sendRequest() {  
    var username=  
        document.getElementById("username").value;  
    var url= "UserRegiterServlet?username="+username;  
    createXMLHttpRequest();  
    XMLHttpRequest.open("GET", url, true);  
    XMLHttpRequest.onreadystatechange =  
        processResponse;  
    XMLHttpRequest.send(null);  
}
```

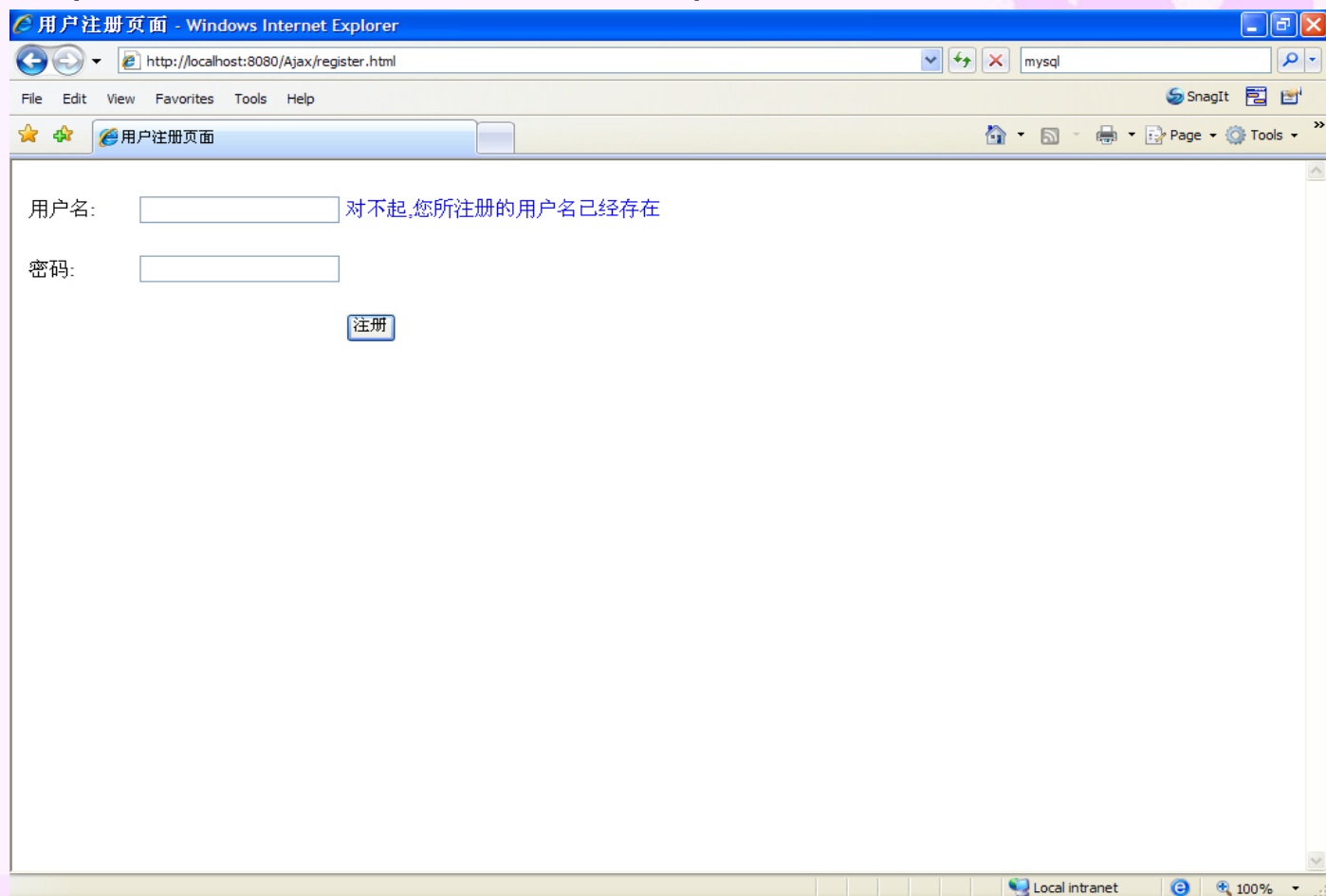
register.html 设计完毕后，继续设计服务器端处理程序 UserRegiterServlet。

```
public void doGet(HttpServletRequest request,
    HttpServletResponse response) throws
    ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String username=request.getParameter("username");
    response.setContentType("text/html; charset=GBK");
    PrintWriter out = response.getWriter();
    try {
        String sql="select * from userinfo where username=?";
```

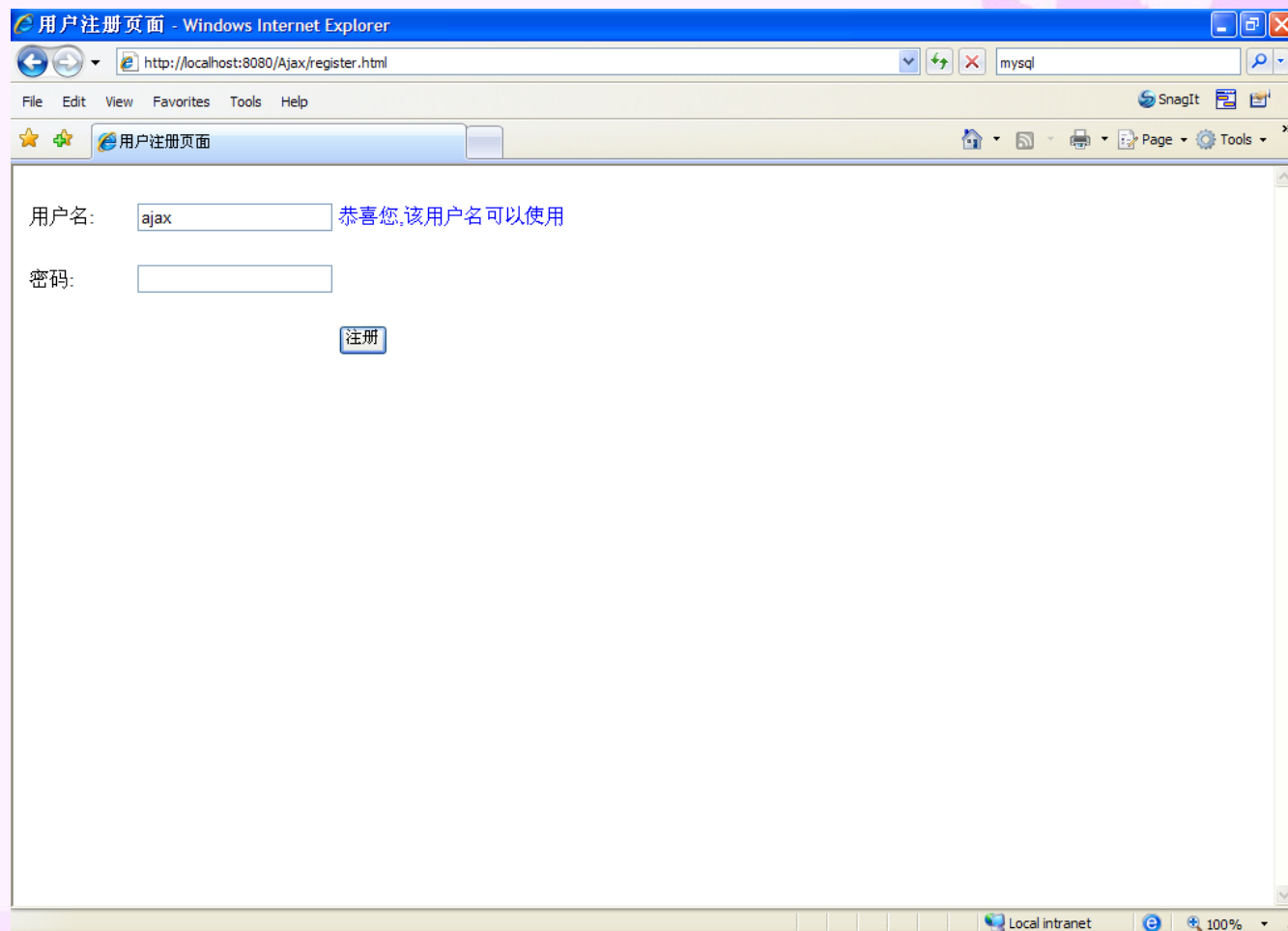


```
DBConnect dbc=new DBConnect(sql);
    dbc.setString(1,username);
    ResultSet rs=dbc.executeQuery();
    if(rs.next())
        out.println(" 对不起，您所注册的用户名已经
存在 ");
    else
        out.println(" 恭喜您，该用户名可以使用 ");
    } catch (Exception e) {
        // TODO: handle exception
    }
    out.close();
}
```

运行该项目，在用户名输入框中输入 Ajaxuser，因为该用户名已经在数据库表中存在，所以当用户名输入框失去焦点后，页面就会提示“对不起，您所注册的用户名已经存在”



如果输入一个用户名表中不存在的用户名该，则页面会提示“恭喜您，该用户名可以使用”，如图 10-13 所示：



10.4.2 智能匹配检索：

智能匹配检索是指用户在输入框中输入文字信息时，系统会自动查找数据库中与该文字相匹配的信息，并提供给用户选择，这样可以大大提高用户的输入效率。实际应用中最典型的例子就是著名的搜索引擎 Google，当用户输入少量的搜索关键字时，Google 会自动提示出相关的搜索关键词，用户可以选择需要的关键词自动完成输入。

本例中需要新建一个 infolist 表，该表有两个字段，其中 infoid 为自增长类型，同时也是该表的主键，另外一个字段 infotitle 表示信息的标题，为 varchar 类型。

	infoid	infotitle
	1	ajax
▶	2	ajaxframework
	3	ajax system
	4	api
	5	google
	6	google suggest

完成数据表的设计后，下面开发本例的程序部分。开发思路如下：

- (1) 用户在表单中的标题文本框里输入信息，触发 keyup 事件，该事件为用户按下并释放按键时触发，在 onkeyup 事件处理中，需要把用户输入的信息提交给 Servlet 程序；
- (2) Servlet 程序从数据库中查找是否有匹配的信息；
- (3) 如果数据库存在匹配的信息，则以 XML 的形式返回给客户端；
- (4) 客户端取得 XML 后进行解析，解析后的数据置于预设的层中，该层的位置在输入框的下方、宽度与输入框相同，并将该层设为可见；
- (5) 单击层中的任一待选标题，触发 onclick 事件处理，把完整的标题写入标题文本框中。

新建页面 suggest.html ，该页面包含一个标题文本框以及一个默认隐含的层，如图 10-15 所示，主要代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=GBK" />
    <title> 输入内容提示 </title>
    <style type="text/css">
        #suggest{
            position:absolute;
            z-index:1;
            border:1px solid black;
            font-size:14px;
        }
        .over{ border:1px solid #999999;
            background:#e8eff7;
            cursor:hand; }
```

```
.out{                                /*out 样式 */
    border:1px solid #FFFFFF;
    background:#FFFFFF;
    cursor:hand;
}

</style>
</head>
<body>
<div id="suggest" style="display:none"></div>
    Google Suggest 效果实例
    <table width="410" border="0" cellpadding="0"
cellspacing="0">
    <tr>
        <td width="63" height="37"> 标题 </td>
        <td width="347"><input name="title" type="text"
id="title" size="50" onKeyUp="sendRequest()"></td>
    </tr>
</table>
</body>
```

由于本例中客户端每次获得的服务器返回的结果不是单个文本信息，而是多个字符串，因此，本例考虑将这多个返回字符串用 XML 的结构来描述，数据格式如下：

```
<response>
  <res>title1</res>
  <res>title2</res>
  .....
</response>
```

在客户端 JavaScript 解析以上 XML 的时候，需要获取的数据是 res 节点的文本值（“title1”、“title2”等）。



在 JavaScript 中，通常采用 DOM 来解析 XML 文件。DOM 是 **Document Object Model** 的缩写，中文名称是文档**对象模型**，是采取一种直观、一致的方式将结构化文档进行模型化处理，形成一棵结构化的文档树，从而提供访问、导航和操作该文档的简易编程接口。

根节点：

DOM 把层次中的每一个对象都称之为节点（NODE），以 HTML 超文本标记语言为例：整个文档的一个根就是 `<html>`，在 DOM 中可以使用 `document.documentElement` 来访问它，它就是整个节点树的根节点（ROOT）。

子节点：

一般意义上的节点，根节点以下最大子节点就是主文档区 `<body>` 了，要访问到 `body` 标签，在脚本中应该用 `document.body`。`body` 区以内所有的文本及 HTML 标签都是文档的节点，分别称为文本节点、元素节点（或者叫标签节点）。

节点之间的关系：

节点之间的关系也是 DOM 中最重要的一个关节，如何正确地引用到节点对象，一定要清楚节点树各个节点的相互描述方式，javascript 脚本就用了各个节点对象的一整套属性和方法去描述另外的节点对象。

节点的绝对引用如表 10-1 所示：

属性	功能
document.documentElement	返回文档的根节点
document.activeElement	返回当前文档中被击活的标签节点
event.fromElement	返回鼠标移出的源节点
event.toElement	返回鼠标移入的源节点
event.srcElement	返回激活事件的源节点

节点的相对引用如表 10-2 所示：（设当前对节点为 node）

属性	功能
node.parentNode	返回父节点
node.childNodes	返回子节点集合（包含文本节点及标签节点）
node.children	返回子标签节点集合
node.textNodes	返回子文本节点集合
node.firstChild	返回第一个子节点
node.lastChild	返回最后一个子节点
node.nextSibling	返回同属下一个节点
node.previousSibling	返回同属上一个节点

节点的各种操作如表 10-3 所示：（设当前的节点为 node）

方法	功能
document.createElement(sNode)	新增标签节点
node.appendChild(oNode)	追加子节点
node.remove() node.removeChild() node.removeNode()	删除节点
node.replaceChild() node.replaceNode() node.swapNode()	替换节点

关于节点信息的判断如表 10-4 所示：

方法	功能
node.contains()	是否包含某节点
node.hasChildNodes()	是否有子节点

在响应函数 processResponse() 中调用 DOM 的 API 来解析 XML，代码如下：

```
var title;                // 标题变量
function processResponse()
{
    if (XMLHttpRequest.readyState == 4)
    {
        if (XMLHttpRequest.status == 200)
        {
            clearSuggest();
            var tips=
XMLHttpRequest.responseXML.getElementsByTagName("res");
            // 查找 XML 中 res 节点，返回 res 的节点数组
            if(tips.length!=0)
            {
```

```
for(i=0;i<tips.length;i++)
    { title= tips[i].firstChild.nodeValue;

// 解析返回的 res 节点数组，取得 <res></res> 间的值存入 title
变量

        createSuggest();}
        displaySuggest();
    }else
    {
        clearSuggest();
        hiddenSuggest();}

    }else
        window.alert(" 您所请求的页面有异常 .");
    }
}
```

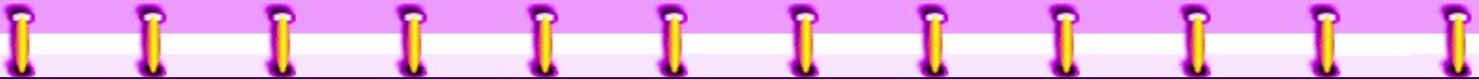
到层上时，层的 CSS 属性设为“over”；鼠标移出时，则设为“out”；鼠标单击时，对标题进行赋值。最后，把生成的代码加到“suggest”层中。

createSuggest() 的具体定义如下：

```
function createSuggest()
{
    var sDiv="<div class='out' onmouseover='mover(this);'
onmouseout='mout(this);' onclick='setSuggest(this)'>" + title + "</div>";
    document.getElementById("suggest").innerHTML+=sDiv;
}
```

以上事件处理中调用的函数 setSuggest、mover 以及 mout 的定义如下：

```
function setSuggest(para)
    document.getElementById("title").value=para.firstChild.nodeValue;
    hiddenSuggest();
}
function mover(para)
    {para.className="over";
    }
function mout(para)
    {para.className="out";}
```



```
function displaySuggest()
{  document.getElementById("suggest").style.display="block";

document.getElementById("suggest").style.width=document.getElementById("title").clientWidth+4;

document.getElementById("suggest").style.left=document.getElementById("title").offsetLeft+73;

document.getElementById("suggest").style.top=document.getElementById("title").style.top+79;

}
```

```
function clearSuggest()
{document.getElementById("suggest").innerHTML="";
}
```

hiddenSuggest() 的作用是隐藏提示层，具体定义如下：

```
function hiddenSuggest()
{
    document.getElementById("suggest").style.display="none";
}
```

完成 processResponse() 及其调用的各个函数后，我们就可以设计函数 sendRequest() 了，代码如下：

```
function sendRequest() {
    var info=document.getElementById("title").value;
    var url="SuggestServlet?info="+info;
    createXMLHttpRequest();
    XMLHttpReq.open("GET", url, true);
    XMLHttpReq.onreadystatechange = processResponse;
    XMLHttpReq.send(null);
}
```

客户端代码设计完毕后，继续设计服务器端处理程序 SuggestServlet。由于本例也是客户端通过 GET 方法发送的请求，因此 SuggestServlet 的 doGet 处理代码如下：

```
request.setCharacterEncoding("UTF-8");  
String info=request.getParameter("info");  
response.setContentType("text/xml;  
charset=UTF-8");  
PrintWriter out = response.getWriter();  
out.println("<response>");  
String sql="select * from infolist where infotitle  
like ?";  
try {  
    DBConnect dbc=new DBConnect(sql);  
    dbc.setString(1,info+"%");  
    ResultSet rs=dbc.executeQuery();
```



```
while (rs.next())
{
    out.println("<res title=\"" + rs.getString("infotitle")
+ ">" + rs.getString("infocontent") + "</res>");
}
rs.close();
dbc.close();
} catch (Exception e) {
    System.out.println(e.getMessage());
}
    out.println("</response>");
    out.close();
}
```

以上代码中，将请求中的参数作为左匹配查询的关键词进行 SQL 语句的构造，并将查询到的标题以 XML 的格式向客户端浏览器进行输出。该 XML 的内容被客户端响应函数 processResponse() 进行处理。

Google Suggest效果实例

标题	a
	ajax
	ajaxframework
	ajaxsystem
	api

选择其中一个选项（如 api），则自动完成标题的输入，无需每一个字母都由用户手工输入。结果如图 10-16 所示：

Google Suggest效果实例

标题	api
----	-----

10.5 第三方 Ajax 高级框架：

1. Prototype

Prototype 与其说是 Ajax 框架，不如说是一个非常强大的 JavaScript 基础库。可以从 <http://www.prototypejs.org/> 免费获得，下载得到的是一个 JavaScript 文件 Prototype.js，该文件包含了基本上所有常见的 JavaScript 操作，并且把这些操作封装成若干个函数，可以非常方便地被调用。

例如：要取得一个 DOM 对象，原先需要通过 `document.getElementById(id)` 来获得，如果采用 Prototype，那么只要用 `$(id)` 就可以轻松地获得。

在对 Ajax 的支持方面，Prototype 也同样很出色，示例代码如下：

```
var myAjax=new Ajax.Request(  
url,                // 请求的服务器程序地址  
{  
    method:'post',    // 请求方法  
    parameters:params, // 传入参数  
    onComplete:showResponse, // 指定回调函数  
    asynchronous:true // 是否异步发送请  
求  
}
```

通过以上的代码，省去了编写复杂冗长的 JavaScript 代码来实现 Ajax。

2.Dojo

Dojo 是一个非常强大的面向对象的 JavaScript 开源工具箱，可以从 <http://dojotoolkit.org/> 免费获得。它比 Prototype 更庞大，而且它不仅仅局限于 Ajax 请求，还包含了许多美观方便的网页客户端控件，如日历、目录树等。只要具备了一定的网页设计基础，利用 Dojo 就可以做出非常美观的页面。

在对 Ajax 的支持方面，Dojo 的使用更加简便、功能更强大，示例代码如下：

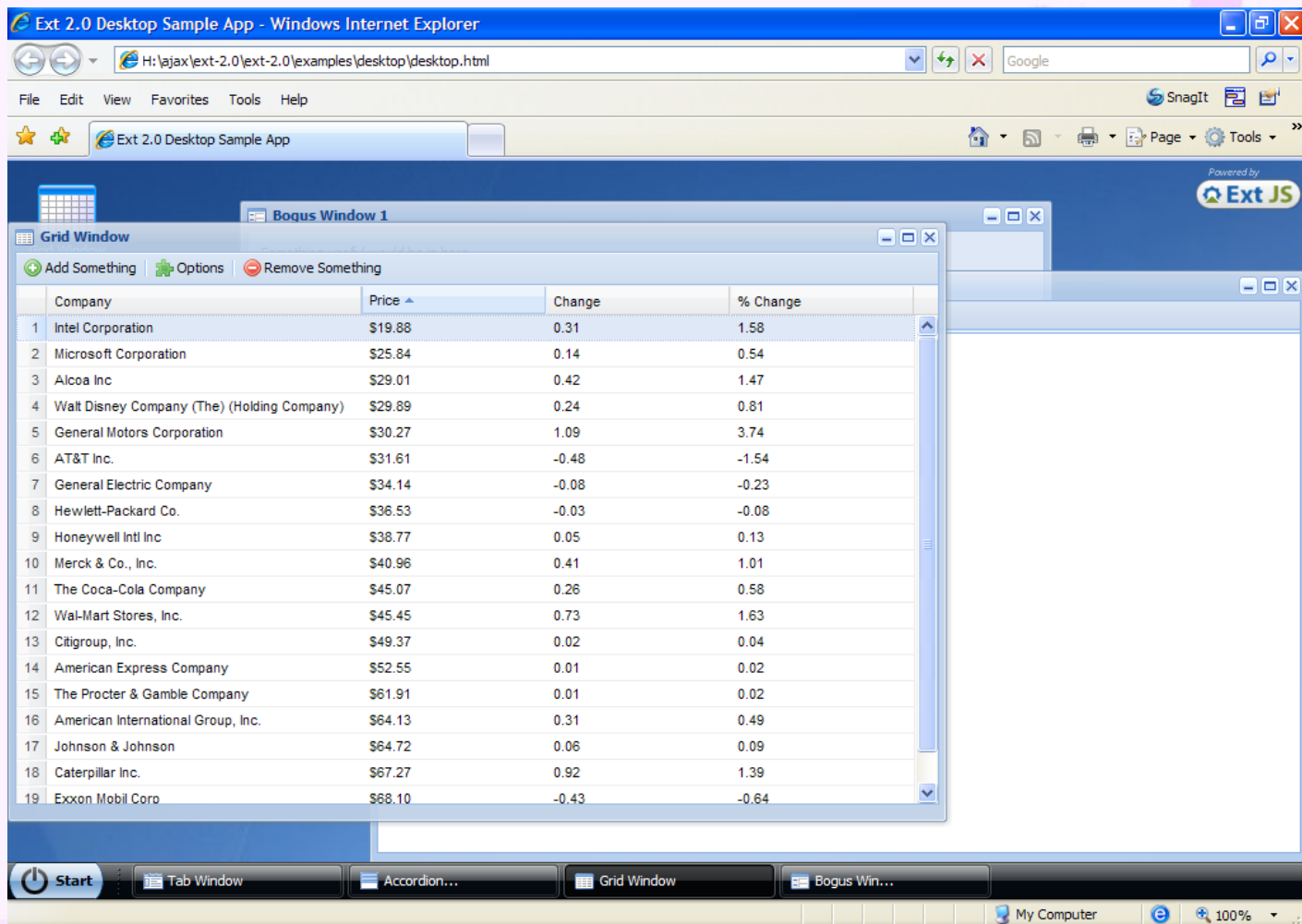
```
dojo.io.bind({  
  url:urlAddress,      // 请求的服务器程序地址  
  method:'POST',      // 请求方法  
  handler:callbackFunction, // 指定回调函  
数  
  content:{param1:value1,param2:value2} // 传入参  
数  
})
```

3.YUI

YUI 全称为 Yahoo! User Interface Library 是由 Yahoo 公司推出的一个 Ajax 开源工具集，读者可以从 <http://developer.yahoo.com/yui/> 下载获得。从 YUI 的全称可以看出，该框架提供了许多 Web 富客户端的控件和操作，它包含的工具具有：拖放操作、页面特效、浏览器事件管理等；UI 控件有日历、提示、面板、对话框、菜单、TabView、TreeView 等。

值得一提的是，在 YUI 的基础上，国外的程序员对其进行了扩展，推出了 YUI-Ext，读者可以从 <http://extjs.com/> 免费获得，该工具包在页面的美观上有了一个很大的改进。

如图 10-17 所示为官方提供的例子：



10.6 本章小节

Web2.0 带给互联网应用更多的交互性，而 Ajax 作为 Web2.0 的典型实现技术之一，在未来对 Web 开发的重要性不言而喻。本章从 Ajax 的定义和工作原理出发，通过多个实例详细介绍了 Ajax 应用的具体实现及开发技巧。从本章内容的学习，读者可以了解 Ajax 本身并不是一种新的技术，而是一些成熟技术结合起来产生的新应用。因此，读者只有在熟练掌握 JavaScript、XML 以及 CSS 等 Ajax 支撑技术的基础之上，才能对 Ajax 的高级应用进行更加深入的学习和研究。

本章习题

- 1、简述 Ajax 技术的定义及其特点。
- 2、简述 Ajax 的工作原理以及与传统处理方式的区别。
- 3、Ajax 技术中的核心对象是什么？如何使用该对象？
- 4、请开发一个与本章智能匹配检索实例相似的 Ajax 应用，要求如下：
 - （1）数据表中包含标题和内容两个字段；
 - （2）不仅自动完成标题的输入，而且要把当前所选标题对应的内容显示在页面上。
- 5、请通过互联网资源的帮助，设计一个基于第三方 Ajax 框架的应用程序（不局限于本章所介绍的 3 种框架）。